Reverse de Bruijn: Utilizing Reverse Peptide Synthesis to Cover All Amino Acid k-mers

Yaron Orenstein

Department of Electrical and Computer Engineering
Ben-Gurion University of the Negev, Beer-Sheva, Israel
yaronore@bgu.ac.il

Abstract. Peptide arrays measure the binding intensity of a specific protein to thousands of amino acid peptides. By using peptides that cover all k-mers, a comprehensive picture of the binding spectrum is obtained. Researchers would like to measure binding to the longest k-mer possible, but are constrained by the number of peptides that can fit into a single microarray. A key challenge is designing a minimum number of peptides that cover all k-mers. Here, we suggest a novel idea to reduce the length of the sequence covering all k-mers by utilizing a unique property of the peptide synthesis process. Since the synthesis can start from both ends of the peptide template, it is enough to cover each k-mer or its reverse, and use the same template twice: in forward and reverse. Then, the computational problem is to generate a minimum length sequence that for each k-mer either contains it or its reverse. We developed an algorithm ReverseCAKE to generate such a sequence. ReverseCAKE runs in time linear in the output size and is guaranteed to produce a sequence that is longer by at most $\Theta(\sqrt{n}\log n)$ characters compared to the optimum n. The obtained saving factor by ReverseCAKE approaches the theoretical lower bound as k increases. In addition, we formulated the problem as an integer linear program and empirically observed that the solutions obtained by ReverseCAKE are near-optimal. Through this work we enable more effective design of peptide microarrays.

Keywords: de Bruijn graph, de Bruijn sequence, peptide array, reverse synthesis, array design.

1 Introduction

Protein-peptide interactions are a central focus of biological research. They play roles in many cellular processes. Some proteins, such as enzymes and antibodies, bind short peptides and by that affect their imminent function. Proteins bind to different peptides with variable affinities. Studying the specificity of protein-peptide binding is a fundamental goal in understanding cellular processes.

Technologies measure the binding intensity of a protein to many peptides (e.g., peptide microarrays [6, 3, 8]). These technologies synthesize a large set of amino acid peptides, and measure the binding intensity of a specific protein to each of these peptides. Some technologies use random peptide sequences [3, 8]. Others use sequences that cover all possible amino acid k-mers [6]. One way to cover all k-mers is to use *de Bruijn sequences*, which are known to be the most compact sequences to cover all k-mers [2, 5]. The length of a de Bruijn sequence of order k over alphabet $|\Sigma|$ is $|\Sigma|^k$, where the amino acid alphabet is of size $|\Sigma| = 20$. Due to the exponential dependency on k and small space on the experimental device, these technologies are limited to a small value of k (e.g. $k = 2$ [6]). Despite the universal and high-throughput nature of these technologies, the data produced are still limited. For many proteins the binding depends on more than two amino acid positions. Covering all k-mers for a greater value of k will lead to improved understanding of peptide interactions.

Here, we utilize for the first time a unique property of amino acid peptide synthesis process to generate smaller peptide libraries. As peptide synthesis can start from both the N-terminus and C-terminus [1], one can save by using this reverse property: if the synthesis starts from both ends, whenever a k-mer is included, its reverse is included as well, and there is no need to cover it again. This brings up the following question: a sequence S is called a *reverse de Bruijn sequence* of order k (RdB sequence for short) if for each k-mer either the k-mer or its reverse are included in S. Can we construct an optimal (minimum length) RdB sequence? Theoretically, if for each k-mer T the sequence S includes either T or its reverse but not both, one could save a factor of nearly 2 compared to the length of a de Bruijn sequence.

Several solutions have been suggested to generate sequence libraries that cover all possible k-mers in the most compact space possible. A de Bruijn sequence is the shortest sequence in which each k-mer appears exactly once. Its length is given by $|\Sigma|^k + k - 1$. De Bruijn sequences were used in protein binding microarrays for $k = 10$ [11]. A reduction of DNA libraries by half was achieved by utilizing the reverse-complementarity property of double-stranded DNA [4, 13, 10]. Other methods produce compact unstructured RNA libraries to measure protein-RNA binding [9, 12]. But, none of those studies considered the property of reverse peptide synthesis, i.e. the need to cover each k-mer by itself or its reverse.

In this study we address the problem of constructing a compact RdB sequence. We take the view point of a sequence as a path in a de Bruijn graph, where an RdB sequence and its reverse are two reverse paths. We first give a lower bound for the length of an RdB sequence. Then, we give a sufficient

and necessary condition for a de Bruijn graph to represent two reverse RdB sequences. As a consequence, we prove that a minimum length RdB cannot achieve the lower bound due to palindromes. We present a linear time near-optimal algorithm, ReverseCAKE (Reverse Covering All K-mErs), to make a de Bruijn graph obtain these properties. Once a de Bruijn graph obtains these properties, a modified Euler tour algorithm can run on it to produce the sequence. Moreover, we formulate the problem as an integer linear program (ILP). We implemented the algorithm and the ILP formulation and we demonstrate the savings they achieve. The results enable saving a factor of almost two compared to using a regular de Bruijn sequence. The code and software are freely available from `https://github.com/yaronore/reverse-de-bruijn`.

2 Preliminaries

A *directed graph* (digraph or simply a graph) $G = (V, E)$ is a set of vertices $V = \{v_1, v_2, \ldots, v_n\}$ and a set of edges $E = \{e_1, e_2, \ldots, e_m\}$. Each edge is an ordered pair of vertices (v_i, v_j), and we say the edge is directed from v_i to v_j. The *indegree* of vertex v is the number of edges entering v. Similarly, the *outdegree* is the number of edges outgoing from v. A vertex is *balanced* if its indegree equals its outdegree. A *path* in a digraph is a sequence of vertices, $v_{i_1}, \ldots, v_{i_k}$, such that for each $1 \leq j < k$ there is an edge $(v_{i_j}, v_{i_{j+1}})$. A *cycle* is a path where $i_1 = i_k$. A digraph is *strongly connected* if for every pair of vertices u, v there exists a path from u to v and a path from v to u.

An *Eulerian tour* through a digraph G is a cycle that traverses all edges in G, such that each edge is traversed exactly once. If a digraph contains an Eulerian tour, we call it *Eulerian*. A digraph is Eulerian if and only if it is strongly connected and all vertices are balanced [14].

A *de Bruijn sequence* of order k over alphabet Σ is a minimum length sequence that covers each k-mer over Σ exactly once. For convenience, we define the *length* of the sequence as the number of k-mers in it. Hence, a sequence of length t contains $t + k - 1$ characters, or t characters if it is cyclic. A de Bruijn sequence has length $|\Sigma|^k$, which is the minimum possible for covering all k-mers.

A *de Bruijn graph* of order k is a digraph in which for every possible k-mer $x_1, \ldots, x_k$ there is a vertex denoted by $[x_1, \ldots, x_k]$. An edge may exist from u to v if $u = [x_1, \ldots, x_k]$ and $v = [x_2, \ldots, x_{k+1}]$. Each edge represents a unique $(k+1)$-mer. For example, the edge (u, v) above represents $(x_1, \ldots, x_{k+1})$. To distinguish vertices from edges, we will use square brackets for vertices. Hence, $(x_1, \ldots, x_{k+1})$ is the edge between $[x_1, \ldots, x_k]$ and $[x_2, \ldots, x_{k+1}]$. In a *complete* de Bruijn graph all possible edges exist, each exactly once. Consequently, for each vertex v the indegree and outdegree are $|\Sigma|$, and the graph is strongly connected. Thus, a complete de Bruijn graph is Eulerian. Any Eulerian tour represents a de Bruijn sequence of order $k + 1$.

The *reverse* of sequence $(x_1, \ldots, x_k)$, denoted $R(x_1, \ldots, x_k)$, is defined as the sequence obtained by reversing the original sequence, i.e. $R(x_1, \ldots, x_k) = (x_k, \ldots, x_1)$. For example, $R(CGAA) = AAGC$. A sequence s is called a *palin-*

dromic sequence, or in short a *palindrome*, if $s = R(s)$. For example, $ACCA$ is a palindrome. An *homomorphic* k-mer is composed of a single letter, e.g. $AA \dots A$.

We define a *reverse de Bruijn sequence* of order k over alphabet Σ (RdB sequence for short) as a sequence such that for each k-mer s, at least one of s and $R(s)$ are in the sequence. Note that unlike a regular de Bruijn sequence, the definition of an RdB sequence does not require minimality. An RdB sequence is *optimal* if it is of minimum length. An RdB sequence is cyclic, and can be easily turned to a linear sequence by appending the first $k-1$ characters.

Given a directed path F in a de Bruijn graph, its *reverse path* is defined as the path R in which each edge (u, v) in F is replaced by the edge $(R(v), R(u))$. For example, for the path $ACG \to CGG \to GGT$, its reverse is $TGG \to GGC \to GCA$ (see Figure 1). We will refer to F and R as forward and reverse paths, respectively.

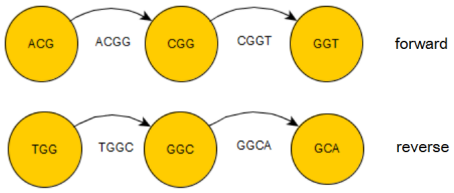

Fig. 1. An illustration of forward and reverse paths (top and bottom, respectively). The forward path traverses the edges in their direction. The corresponding reverse path traverses the reverse edges in reverse direction.

3 Results

3.1 A Lower Bound for the Length of an RdB Sequence

We derive a lower bound for the length of an RdB sequence from k-mer counts.

Proposition 1. *Denote $n(k)$ the length of an optimal RdB sequence of order k.*

$$n(k) \geq \frac{1}{2} \cdot (|\Sigma|^k + |\Sigma|^{\lfloor (k+1)/2 \rfloor}) \tag{1}$$

Proof. We derive the lower bound by counting palindromic and non-palindromic edges. It depends on the number of k-mers that are palindromes, since each palindrome must be represented by itself, while each non-palindromic k-mer can be represented by either itself or its reverse. For even k the first $\frac{k}{2}$ characters define the last $\frac{k}{2}$ characters of a palindrome. For odd k, the first $\frac{k-1}{2}$ characters define the last $\frac{k-1}{2}$, and the middle character can be any letter. Hence, there are exactly $|\Sigma|^{\lfloor (k+1)/2 \rfloor}$ different palindromes. In total, counting all palindromes and half of all non-palindromes gives $n(k) \geq \frac{1}{2} \cdot (|\Sigma|^k - |\Sigma|^{\lfloor (k+1)/2 \rfloor}) + |\Sigma|^{\lfloor (k+1)/2 \rfloor}$.

3.2 A de Bruijn Graph Representing Two Reverse RdB Sequences

We give a sufficient and necessary condition for a de Bruijn graph to represent two reverse RdB sequences. This will be useful to prove that no RdB can achieve the lower bound. It will also be relevant for a de Bruijn graph edge augmentation we show below, as it will make a complete de Bruijn graph obtain these properties. We take the viewpoint of a sequence represented as a path in a de Bruijn graph. We first prove the following lemma:

Lemma 1. *For every incoming non-homomorphic edge e into a palindromic vertex, there is a unique outgoing edge e' such that $e' = R(e)$.*

Proof. Denote the vertex label as $v = [x_1, \ldots, x_k]$, which is equal to $[x_k, \ldots, x_1]$ as it is a palindromic vertex. Denote an incoming edge by $e = (y, x_1, \ldots, x_k)$. Its reverse is $R(e) = (x_k, \ldots, x_1, y)$, which is an outgoing edge from v. $e = R(e)$ if and only if e is homomorphic.

Theorem 1. *The set of edges of de Bruijn graph G represents two reverse RdB sequences $\iff$ de Bruijn graph G has the following properties:*

1. *All vertices in G are balanced.*
2. *G is strongly connected.*
3. *There is a perfect matching of edges and their reverse in G.*
4. *All palindromic vertices in G have an even in and outdegree (disregarding homomorphic edges).*

Proof. $\rightarrow$ Each k-mer in a sequence is an edge in the graph. As an RdB sequence and its reverse are cyclic each vertex is entered and exited the same number of times, and it follows that the vertices are balanced. As an RdB sequence and its reverse cover all k-mers, all possible edges exist and it follows that the graph is strongly connected. The edges of the paths representing the RdB sequence and its reverse are in prefect matching of reverse edges with each other by definition of forward and reverse paths. Last, by Lemma 1 each palindromic vertex is entered an exited by both paths at the same time. Thus, it must be entered and exited an even number of times (disregarding homomorphic edges whose traversal does not change the paths' location).

$\leftarrow$ Given that de Bruijn graph G has the four properties, it has two reverse paths that cover all of its edges. The paths enter and exit each vertex the same number of times as the vertices are balanced. The paths cover together all edges as the graph is strongly connected. The perfect matching is necessary to have two reverse paths in the graph. Last, by Lemma 1 when the paths enter the same vertex (palindromic vertices) they never reach a dead-end as there is an even number of outgoing and incoming edges (disregarding homomorphic edges whose traversal does not change the paths' location) .

As a consequence, no RdB sequence can achieve the lower bound:

Corollary 1. *There is no RdB sequence that achieves the lower bound.*

Proof. Assume, contrary to the claim, that there is a reverse Bruijn sequence that achieves the lower bound. Thus, the sequence and its reverse path are two edge-disjoint paths in an augmented de Bruijn graph, where each original palindromic edge is doubled and all other edges appear only once. The augmented graph has vertices with unequal indegree and outdegree due to the augmentation of palindromic edges, contradicting Therorem 1.

Give a graph with the listed properties we can apply an algorithm to find two reverse paths that cover all edges. The algorithm is based on a modification of the Euler tour algorithm [14] run on an augmented de Bruijn graph. The algorithm for generating the sequence will work on an augmented de Bruijn graph of order $k-1$. We previously presented it [10] and repeat it here for sake of clarity.

Algorithm 1 Find forward and reverse paths that cover all edges in an augmented de Bruijn graph $G = (V, E)$ of order $k-1$.

1. Initially all edges are unmarked, $\mathcal{F} = \mathcal{R} = \emptyset$, and $A = \{u\}$, an arbitrary vertex.
2. While $A \neq \emptyset$ do
3. $F = R = \emptyset$. Pick any starting vertex $v = [x_1, \ldots, x_{k-1}]$ from A.
4. While there exists an unmarked edge $e = (x_1, \ldots, x_k)$ outgoing from v do
5. Append e to F. Prepend $R(e)$ to R.
6. Mark e and $R(e)$.
7. Set $v = [x_2, \ldots, x_k]$; $A = A \cup \{v\}$.
8. Remove v from A.
9. If $F \neq \emptyset$, add F to $\mathcal{F}$; add R to $\mathcal{R}$;
10. Merge the cycles in $\mathcal{F}$ to obtain a single forward path. Do the same for $\mathcal{R}$.

Theorem 2. *Algorithm 1 returns in $O(|V|)$ time forward and reverse paths that cover together all edges of the augmented graph and represent two RdB sequences.*

The algorithm and its proof are similar to that of a modified Euler tour algorithm we previously presented [10] (and will not be repeated here). We first show that if the forward path F reaches a dead-end, then so does the reverse path R, and in that case a cycle is closed (note that each pair F, R constructed in steps 4-7 are reverse paths by the way they are constructed). Then, we show that the cycles in $\mathcal{F}$ can be merged into one cycle. Third, we deduce that a strongly connected component is covered by $\mathcal{F}$ and $\mathcal{R}$. Last, we conclude that $\mathcal{F}$ and $\mathcal{R}$ cover all edges, since there is only one strongly connected component in any de Bruijn graph. The only difference between the proofs is the case where both paths enter the same vertex simultaneously and reach a dead-end.

Lemma 2. *If the forward traversal reaches a dead-end at a palindromic vertex, then so does the reverse at the same vertex. Both paths close a cycle in this case.*

Proof. Recall that when F reaches a palindromic vertex R must reach it a well, and this is the only case where both paths reach a vertex together. By Lemma 1 a

palindromic vertex has even in and outdegrees (excluding homomorphic edges). Denote by $(x_1, \ldots, x_k)$ an incoming edge used by F. Then, the reverse outgoing edge, which is traversed by R, is $(x_k, \ldots, x_1)$. As it is a palindromic vertex, or equivalently from the fact that both reach the vertex simultaneously, we get that $[x_2, \ldots, x_k] = [x_k, \ldots, x_2]$. It follows that in all traversals of this vertex F and R reach the vertex simultaneously. Hence, when F reaches a dead-end, all incoming and outgoing edges were already traversed, and they are all of the form $(a, x_2, \ldots, x_n)$ and $(x_n, \ldots, x_2, a)$, $\forall a \in \Sigma$. For each traversal of such pair of incoming edges, a pair of outgoing edges is traversed. Thus, if the traversal ends at the vertex, it must be that the traversal started from that vertex (otherwise, there would have been unmarked outgoing edges to traverse). In other words, both paths close a cycle in this case.

3.3 Constructing a Near-Optimal RdB Sequence in Linear Time

In this approach, for each palindromic edge, we add to a complete de Bruijn graph all possible cyclic shifts of it. More formally, for even k let $k = 2l$. For the palindrome $e = (x_1, \ldots, x_l, x_l, \ldots, x_1)$ we add k edges corresponding to all possible cyclic shifts of e. Similarly, for odd k let $k = 2l + 1$ we add all cyclic shifts of $(x_1, \ldots, x_l, x_{l+1}, x_l, \ldots, x_1)$. Obviously, since these edges form a cycle, all vertices remain balanced. The added edges match in reverse pairs. For each edge that represents the cyclic shift starting at position i, for $1 < i < \lfloor (k+1)/2 \rfloor$, the matching edge starts at $k+2-i$. Hence, a perfect matching exists after adding the new cycles. For even k, unless the k-mer is homomorphic, this cycle contains two edges that are palindromes, $(x_1, \ldots, x_l, x_l, \ldots, x_1)$ and $(x_l, \ldots, x_1, x_1, \ldots, x_l)$, so only one cycle is added for both, and the cycle doubles both palindromic edges. In total, during the edge augmentation process, for each palindromic k-mer we add at most k edges. For example, for the palindromes $AGGA$ and $GAAG$ we add $AGGA$, $GGAA$, $GAAG$ and $AAGG$ (see Figure 2). The added edges $GGAA$ and $AAGG$ match each other as a reverse edge pair. The added palindromes match the original edges in the graph. The resulting augmented graph contains at most $|\Sigma|^k + k \cdot |\Sigma|^{\lfloor (k+1)/2 \rfloor}$ edges, where the first term is the number of edges in the original de Bruijn graph and the second is k edges for each palindrome.

In some cases, the number of added edges can be reduced. If the palindrome $(x_1, \ldots, x_k)$ is periodic, then the number of cyclic shifts needed to return to the original k-mer is the length of the period. For example, the period of $ACCAACCA$ is 4. Only four edges suffice in this case, the edges $ACCAACCA$, $CCAACCAA$, $CAACCAAC$ and $AACCAACC$. So, each periodic palindrome requires an addition of the number of edges equal to the length of its period. Hence, a smaller augmented graph and a shorter RdB sequence can be obtained by considering the different possible periods.

At the end of the above augmentation, an additional augmentation is required to palindromic vertices with odd degrees. All palindromic vertices must have an even in and outdegrees (disregarding homomorphic edges) by Lemma 1. To make sure all palindromic vertices have even degrees following the above augmentation, all palindromic vertices with odd degrees are matched in pairs. Then, $k - 1$

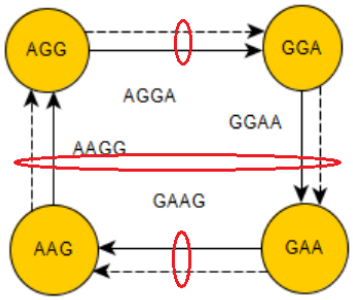

Fig. 2. A cycle and edge matching. For the pair of palindromes $AGGA$ and $GAAG$, all cyclic shifts of these palindromes are added once (dashed edges). In the matching, palindromic edges in the original cycle are paired with their added copies (encircled by small red ovals). Other non-palindromic added edges are paired (encircled by a large red oval).

edges connecting them in a cycle are added. This augmentation preserves degree balance, graph connectivity, perfect matching of reverse edges and does not affect degree of other palindromic vertices. Since there are at most $|\Sigma|^{\lfloor k/2 \rfloor}$ palindromic vertices, in this process at most $(k-1)|\Sigma|^{\lfloor k/2 \rfloor}$ edges are added.

Algorithm 1 produces two sequences, forward and reverse, each of which is an RdB sequence (Figure 3), in time linear in the size of the graph [10]. The length of each of the produced sequences is the number of edges divided by two. For each palindromic edge at most k edges were added. For each palindromic vertex 0 or $k-1$ edges were added. So, the total length of the sequence is bounded by $(|\Sigma|^k + k|\Sigma|^{\lfloor (k+1)/2 \rfloor} + (k-1)|\Sigma|^{\lfloor k/2 \rfloor})/2$. This is an addition of $\Theta(\sqrt{L}\log(L))$ characters, where L denotes the lower bound in Proposition 1 for an RdB sequence of order k. We call the augmentation process followed by Algorithm 1 ReverseCAKE.

Figure 4A (and Table 1 in the Appendix) show the results of ReverseCAKE for different values of k. As we can see, the sequence obtained by ReverseCAKE is of length nearly half that of the original de Bruijn sequence. For example, for $k = 4$ and amino acid alphabet, it is within 1 percent of $20^4/2$ and within 220 characters from the lower bound.

3.4 Integer Linear Programming Formulation

We present an ILP formulation to calculate the minimum length RdB sequence. There are $|\Sigma|^k$ integer variables X_i. Each X_i corresponds to the number of times the k-mer occurs in the sequence.

As we aim for the shortest sequence, the objective function is

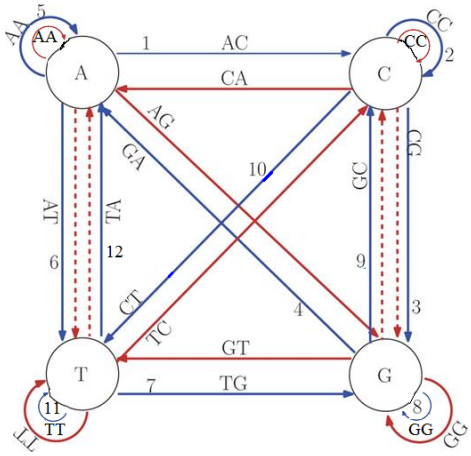

Fig. 3. An augmented de Bruijn graph of order 1 and an example of forward and reverse paths in it. Palindromic edges AA, CC, GG and TT were added first as cyclic shifts of all palindromes. Then, dashed edges AT, TA, CG and GC were added to turn odd degree palindromic vertices to even degree. The blue and brown paths represent the forward and reverse paths, respectively. Numbers on edges are the order of the edges in the forward path. The sequences are ACCGATGGCTTA and ATTCGGTAGCCA for forward and reverse paths, respectively.

$$\min \sum_{i=1}^{|\Sigma|^k} X_i \tag{2}$$

The first constraint is the coverage constraint, which requires that all k-mers occur in the sequence as themselves or their reverse. Let $R(i)$ denote the reverse of k-mer i, where we use the integer representation of a k-mer as a number in radix 4.

$$X_i + X_{R(i)} \geq 1 \qquad 1 \leq i \leq |\Sigma|^k \tag{3}$$

The second constraint guarantees that the k-mer occurrences can form a (cyclic) sequence. We require that for each $(k-1)$-mer the number of k-mers with that $(k-1)$-mer in their suffix is equal to the number of k-mers with that $(k-1)$-mer in their prefix (equivalent to a flow conservation constraint). Denote $p_x(i)$ and $s_x(i)$ the x-long prefix and suffix of i, respectively.

$$\sum_{s_{k-1}(i')=i} X_{i'} = \sum_{p_{k-1}(i')=i} X_{i'} \qquad 1 \leq i \leq |\Sigma|^{k-1} \tag{4}$$

We compared the memory usage and runtime of ReverseCAKE and the ILP solver. The results are summarized in Figures 4B and 4C (and Table 2 in the Appendix). We used Gurobi ILP 7.5.2 solve to solve the ILP formulation [7].

Running times and memory usages were benchmarked on a single CPU of a 20-CPU Intel Xeon E5-2650 (2.3GHz) machine with 384GB 2133MHz RAM. In all runs reported, the ILP solver reached an optimal solution. As expected, the ILP solver requires much more time and memory. Our linear time algorithm produces a sequence that is only negligibly longer, but in much shorter times and using less memory.

4 Summary and discussion

We studied the problem of constructing a minimum length sequence that covers each k-mer by itself or its reverse. The problem has applications in constructing dense amino acid peptide arrays for measuring protein-peptide interactions [6, 3, 8]. Using on our solution researchers will be able to improve the utilization of peptide arrays in high-throughput, universal and unbiased measurement of peptide interactions.

The problem is challenging due to palindromes, which are the reverse of themselves and must appear in any sequence. We present a linear time near-optimal algorithm ReverseCAKE to solve it. In practice, the length of the sequence produced by the algorithm nearly halves the total length of the sequence. It is very close to the optimum as empirically shown by the integer linear programming solutions. We believe that our results are of theoretical interest, and are applicable in current and future technologies that require complete coverage of amino acid k-mers under harsh space constraints.

Our study raises several open questions. Can one construct an optimal RdB sequence in polynomial time? Second, what is the number of different optimal RdB sequences? Third, can one design an optimal RdB sequence with improved coverage of gapped k-mers in cases of gapped peptide interactions? Fourth, is there a closed formula for the length of an optimal RdB sequence? Finally, in current technologies, the de Bruijn (or RdB) sequence is cut into probes of length p with overlap $k-1$. There is no constraint that forces these probes to come from a single sequence. What is the minimum number of sequences of length p that cover all k-mers, each k-mer by itself or its reverse? Since our solution for an RdB sequence covers a few k-mers more than once (as shown by the gap between the theoretical lower bound and our solutions), a direct design of probe sequences of length p might be able to reduce the number of probes needed to cover all k-mers.

Appendix

References

1. Benoiton, N.L.: Chemistry of peptide synthesis. CRC Press (2016)
2. Berger, M.F., Philippakis, A.A., Qureshi, A.M., He, F.S., Estep 3rd, P.W., Bulyk, M.L.: Compact, universal DNA microarrays to comprehensively determine transcription-factor binding site specificities. Nature biotechnology 24(11), 1429 (2006)

3. Buus, S., Rockberg, J., Forsström, B., Nilsson, P., Uhlen, M., Schafer-Nielsen, C.: High-resolution mapping of linear antibody epitopes using ultrahigh-density peptide microarrays. Molecular & Cellular Proteomics 11(12), 1790–1800 (2012)
4. D'Addario, M., Kriege, N., Rahmann, S.: Designing q-unique DNA sequences with integer linear programs and Euler tours in de Bruijn graphs. In: OASIcs-OpenAccess Series in Informatics. vol. 26. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2012)
5. Fordyce, P.M., Gerber, D., Tran, D., Zheng, J., Li, H., DeRisi, J.L., Quake, S.R.: De novo identification and biophysical characterization of transcription-factor binding sites with microfluidic affinity analysis. Nature biotechnology 28(9), 970–975 (2010)
6. Gurard-Levin, Z.A., Kilian, K.A., Kim, J., Bahr, K., Mrksich, M.: Peptide arrays identify isoform-selective substrates for profiling endogenous lysine deacetylase activity. ACS chemical biology 5(9), 863–873 (2010)
7. Gurobi Optimization, I.: Gurobi optimizer reference manual (2016), `http://www.gurobi.com`
8. Halperin, R.F., Stafford, P., Johnston, S.A.: Exploring antibody recognition of sequence space through random-sequence peptide microarrays. Molecular & Cellular Proteomics 10(3), M110–000786 (2011)
9. Orenstein, Y., Berger, B.: Efficient design of compact unstructured RNA libraries covering all k-mers. Journal of Computational Biology 23(2), 67 (2016)
10. Orenstein, Y., Shamir, R.: Design of shortest double-stranded DNA sequences covering all k-mers with applications to protein-binding microarrays and synthetic enhancers. Bioinformatics 29(13), i71–i79 (2013)
11. Philippakis, A.A., Qureshi, A.M., Berger, M.F., Bulyk, M.L.: Design of compact, universal DNA microarrays for protein binding microarray experiments. Journal of Computational Biology 15(7), 655–665 (2008)
12. Ray, D., Kazan, H., Cook, K.B., Weirauch, M.T., Najafabadi, H.S., Li, X., Gueroussov, S., Albu, M., Zheng, H., Yang, A., et al.: A compendium of RNA-binding motifs for decoding gene regulation. Nature 499(7457), 172 (2013)
13. Smith, R.P., Riesenfeld, S.J., Holloway, A.K., Li, Q., Murphy, K.K., Feliciano, N.M., Orecchia, L., Oksenberg, N., Pollard, K.S., Ahituv, N.: A compact, in vivo screen of all 6-mers reveals drivers of tissue-specific expression and guides synthetic regulatory element design. Genome biology 14(7), 1 (2013)
14. West, D.B., et al.: Introduction to graph theory, vol. 2. Prentice hall Upper Saddle River (2001)

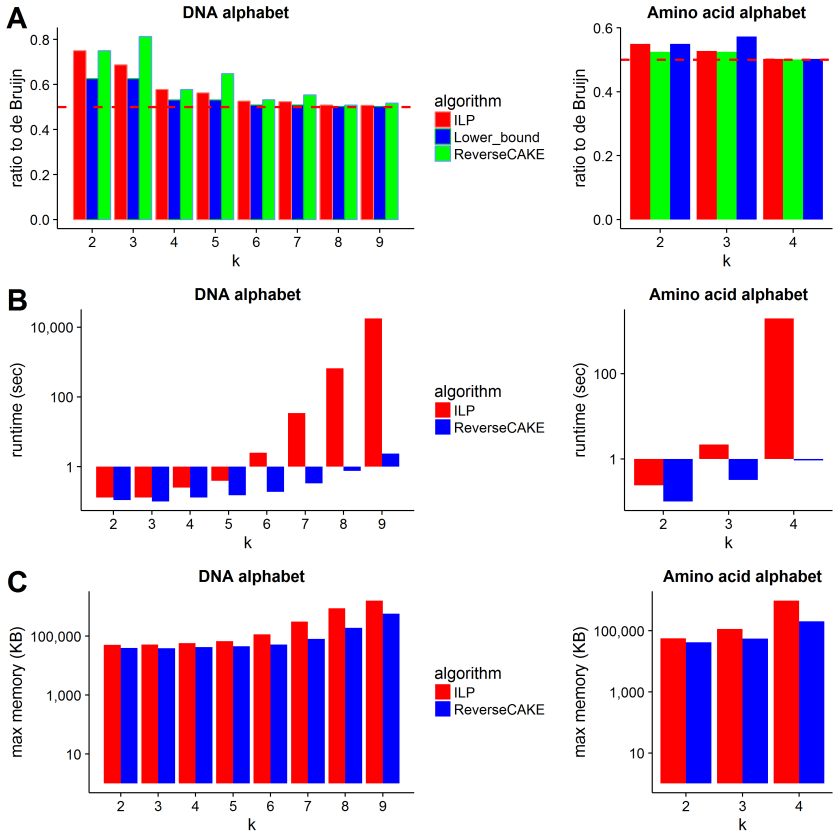

Fig. 4. Results and performance of ReverseCAKE and the ILP solver. (A) Results of sequence lengths are portrayed as ratios to an original de Bruijn sequence ($|\Sigma|^k$). The lower bound is from Proposition 1. The dashed red line is at half. (B,C) Runtimes and maximum memory usage of the algorithms, respectively. Y-axis is on a log-scale.

Table 1. Lengths of reverse de Bruijn sequences produced by ReverseCAKE and an ILP solver. The columns are organized as follows: (i) the alphabet, where aa stands for amino acid; (ii) the length of a regular de Bruijn sequence that does not exploit reverse peptide synthesis; (iii) the lower bound on RdB sequence length (Proposition 1); (iv-v) the lengths of the sequence computed by ReverseCAKE (Section 3.3) and an ILP solver that reached an optimal solution (Section 3.4); (vi) the saving factor is the ratio between an optimal RdB sequence and a de Bruijn sequence.

k	alphabet	de Bruijn	lower bound	ReverseCAKE	ILP (optimal)	saving factor
2	DNA	16	10	12	12	0.75
3	DNA	64	40	52	44	0.69
4	DNA	256	136	148	148	0.58
5	DNA	1,024	544	664	576	0.56
6	DNA	4,096	2,080	2,180	2,156	0.53
7	DNA	16,384	8,320	9,076	8,584	0.52
8	DNA	65,536	32,896	33,276	33,276	0.51
9	DNA	262,144	131,584	135,628	133,064	0.51
2	aa	400	210	220	220	0.55
3	aa	8,000	4,200	4,580	4,220	0.53
4	aa	160,000	80,200	80,420	80,420	0.50

Table 2. Performance evaluation of ReverseCAKE and an ILP solver. The runtime and maximum memory usage are reported in seconds (sec) and kilobytes (KB).

k	alphabet	ReverseCAKE (sec)	ReverseCAKE (KB)	ILP (sec)	ILP (KB)
2	DNA	0.11	39,456	0.13	50,608
3	DNA	0.10	38,964	0.13	52,496
4	DNA	0.13	42,456	0.25	58,308
5	DNA	0.15	45,928	0.39	67,236
6	DNA	0.19	52,036	2.50	114,118
7	DNA	0.33	81,476	34.43	310,752
8	DNA	0.76	191,828	665.15	875,740
9	DNA	2.38	579,220	17,592.82	1,622,996
2	aa	0.10	41,804	0.24	56,800
3	aa	0.32	55,336	2.18	115,272
4	aa	0.92	203,548	1,963.64	982,124